	Type	Hits	Search Text
1	BRS	344	interface$3 with customiz$7 with port$5
2	BRS	24	interface$3 with customiz$7 with port$5 and (platform adj independen$5)
3	BRS	118	interface$3 with customiz$7 with (port$5 not portal$3 not portable$2)
4	BRS	41	interface$3 same customiz$7 with (port$5 not portal$3 not portable$2 not portion$2)
5	BRS	420	interface$3 with (port$5 not portal$3 not portable$2 not portion$2) with application$3
6	BRS	6	interface$3 with (port$5 not portal$3 not portable$2 not portion$2) with application$3 with setting$3
7	BRS	48	interface$3 same (port$5 not portal$3 not portable$2 not portion$2) same application$3 same setting$3
8	BRS	8	interface$3 same ((port$5 not portal$3 not portable$2 not portion$2) near3 setting$3) same application$3
9	BRS	0	((port$5 not portal$3 not portable$2 not portion$2) near3 setting$3) same ((platform adj independen$5) with
10	BRS	59	((port$5 not portal$3 not portable$2 not portion$2) near3 setting$3) same application$3
11	BRS	30	((port$5 not portal$3 not portable$2 not portion$2) near3 setting$3) with application$3
12	BRS	117	((port$5 not portal$3 not portable$2 not portion$2) near3 set$6) with application$3
13	IS&R	1	("5991535").PN.
14	BRS	430	interfac$5 with (port$5 not portal$3 not portable$2 not portion$2) with application$3

	Type	Hits	Search Text
15	BRS	2	interfac$5 with (port$5 not portal$3 not portable$2 not portion$2) with application$3 and 717/$.ccls.
16	BRS	88	interfac$5 with (port$5 not portal$3 not portable$2 not portion$2) with application$3 and 709/$.ccls.
17	BRS	2	interfac$5 with (port$5 not portal$3 not portable$2 not portion$2) with application$3 with virtual adj machine$3
18	BRS	18	(port$5 not portal$3 not portable$2 not portion$2) with application$4 with customiz$7
19	BRS	353	(port$5 not portal$3 not portable$2 not portion$2) with application$4 with set$6
20	BRS	2	(port$5 not portal$3 not portable$2 not portion$2) with application$4 with set$6 and 717/$.ccls.
21	BRS	259	connector adj object$3
22	BRS	8	(connector adj object$3) and 717/$.ccls.
23	BRS	88	(connection adj object$3) and 717/$.ccls.
24	BRS	7	gross.xa. and 717/$.ccls.
25	BRS	267	(port$5 not portal$3 not portable$2 not portion$2) with application$3 with configur$6
26	BRS	6	(port$5 not portal$3 not portable$2 not portion$2) with application$3 with configur$6 and 717/$.ccls.
27	BRS	62	(port$5 not portal$3 not portable$2 not portion$2) with application$3 with configur$6 and 709/$.ccls.

	Type	Hits	Search Text
28	BRS	14	(port$5 not portal$3 not portable$2 not portion$2) near2 configur$6 with application$3 and 709/$.ccls.
29	BRS	69	(port$5 not portal$3 not portable$2 not portion$2) near2 configur$6 with application$3
30	BRS	3	("5787245"|"6314448"|"6393497").PN.
31	BRS	1363	interfac$5 with peripheral$3 with application$3
32	BRS	26	interfac$5 with peripheral$3 with application$3 and
33	BRS	42	interfac$5 with peripheral$3 with application$3 with access$5 with control$5
34	BRS	1	interfac$5 with peripheral$3 with application$3 same (includ$5 with exclud$5)
35	BRS	380	interfac$5 with peripheral$3 with application$3 with (includ$5 or exclud$5)
36	BRS	38	(interfac$5 near2 application$3) with (peripheral$3 near2 (includ$5 or exclud$5))
37	BRS	65	(interfac$5 near2 application$3) with (memor$6 near3 allocat$6)
38	BRS	7	(interfac$5 near3 application$3) with plug$2 with socket$2
39	BRS	1449	(interfac$5 near3 application$3) with librar$6
40	BRS	32	(interfac$5 near3 application$3) with librar$6 same (virtual adj machine$2)
41	BRS	0	(java adj class adj librar$6) with (different near2 (architect$5 or platform$5))
42	BRS	7	(java adj class adj librar$6) with ((architect$5 or platform$5))
43	BRS	72	(interfac$5 near3 application$3) with librar$6 with ((architect$5 or platform$5))

	Type	Hits	Search Text
44	IS&R	1	("6697819").PN.

Microsoft sheds its Java skin -- again

Another series of changes to Microsoft's Java implementation has Sun officials hot on the trail of incompatibilities

By John Zukowski

The current state of affairs with regard to Microsoft's Java environments is changing again.

Microsoft recently released two new versions of its command-line SDK for Java environments: version 2.02, an upgrade from its 2.01 release, and 3.0, a pre-release version. (At the time of publication, Microsoft had not yet revealed the date the full releases will be available.) In addition, Microsoft's oft-maligned Visual J++ (VJ++) 6.0 is newly available in its second beta release (available for general release around mid-September), and the runtime environment for the company's Internet Explorer (IE) Web browser includes a new service pack (SP1) that users can add on to the 4.01 release. (SP1 gives IE4 access to the new Windows Foundation Classes library, first introduced with VJ++ 6.0.)

With all these changes, Sun's Carla Schroer, senior engineering manager of the Java Compatibility Suite, is taking another look at just how compatible Microsoft's latest implementation is and she isn't being shy about telling people what she's found. And, in the spirit of providing an updated Pitfalls article, we're telling it as it is, too.

Schroer reports that the compatibility of Microsoft's Java releases is a "moving target. With each new product, there are differences." And these differences need to be analyzed to be understood. This latest release, the 3.0 pre-release, includes a cleaned-up version of the mislabeled class libraries that Sun previously complained about. The good news: the public AWT peer classes in the java.awt package as well as the added public methods and instance variables have been removed. The bad news: if you created any Java programs that used these capabilities, they'll no longer work. More good news: you will no longer be able to create programs that access non-standard classes, methods, and variables within the java.* packages.

Although the cleaned-up Core API is a good thing, Schroer feels the remaining incompatibilities -- the lack of JNI and complete RMI support, as well as added keywords and compiler directives -- are still important.

JNI is the native code interface used to access platform-specific capabilities, like a serial port or a microphone, for things that aren't available yet through the Core API. The goal of JNI is to permit developers to provide a single set of native libraries for every Java implementation on a specific

platform. Although once you decide to use JNI, you lock your Java program to the platforms in which you provide native libraries, the lack of JNI support in Microsoft products requires native-library vendors to maintain multiple libraries for integration into different Java virtual machines on the same platform.

When it comes to RMI, the remote method invocation library for executing Java code directly within other Java virtual machines, Schroer reports that while Microsoft has provided the RMI classes separately from all its products, it doesn't provide the rmic compiler. According to Schroer, the *Technology License and Distribution Agreement* requires Microsoft, where it provides a development environment, to deliver one that is *complete.* The lack of the rmic compiler makes both the SDK and J++ environments incomplete and, thus, incompatible and in violation of the agreement. Bill Dunlap, the Visual J++ product manager at Microsoft, disagrees with this assessment.

Finally, says Schroer, the added keywords and compiler directives create Java source code that is not compiler and runtime-environment independent. This means that Microsoft's additions "do not just create a runtime tie, but a development environment tie, too." If you use them, you must forever use Microsoft's development tools and runtime environment. While support from Microsoft for the added keywords (`multicast` and `delegate`) is specific to Microsoft's Java VM, nothing in the implementation is specific to Microsoft's VM. Someone could reimplement the functionality using Java's reflection capabilities for other JVMs. While this approach would be slower than Microsoft's native library implementation, it would work.

The compiler directives issue seems to be more of a true compatibility problem; non-descriptive attributes are placed in class files that only Microsoft's Java VM understands. By compiler directives, I am referring to the `@dll` commands in javadoc-style comments, like these:

```
/** @dll.structmap([type=FIXEDARRAY, size=2]) */
/** @dll.struct(pack=1, auto) */
/** @dll.import("GDI32",auto,setLastError) */
/** @dll.import("KERNEL32",auto,entrypoint="GetLargestConsoleWindowSize") */
```

According to Schroer, adding attributes to class files is perfectly legal, as long as they're only descriptive. However, adding functionality behind those attributes is in violation. As an example, Schroer described the `@dll.import` directive. When the Microsoft compiler finds this directive in the source, it adds an attribute to the class file to load the specified dynamic library. However, the standard Java way to load a library is with the `System.loadLibrary()` method. Because non-Microsoft Java VMs will not recognize the attribute added by the `@dll.import` directive, the necessary library will not load and the program will not work correctly. Hence the claim of incompatibility.

Schroer is quick to point out that these claims result in "no lawsuit change." It's just that there are "different incompatibilities with each product" release. Which, I'm sure, makes compatibility testing very difficult.

And how does Microsoft feel about this? According to the whitepaper *Microsoft's Java Strategy: Helping Java Developers Succeed* (April 1997), "Microsoft's Java strategy calls for it to deliver both the power for real, cross-platform applications and the choice to create great Windows-based applications using Java that take full advantage of customers' hardware and software investments."

While these latest changes let developers better create cross-platform solutions (now that the Core Java APIs are back to Sun's defined Core), a later statement in the whitepaper seems to be at jeopardy: "To do this, Microsoft remains committed to best-of-breed Java support that's *fully*

compatible with existing Java tools and code." Fully compatible? I don't think so. The added keywords and compiler directives are clearly not fully compatible with existing tools and code.

Welcome to the world of proprietary Java.

About the author

John Zukowski is a software mage with MageLang Institute, author of *Java AWT Reference* from O'Reilly & Associates and *Borland's JBuilder: No experience required* from Sybex, as well as the Focus on Java guide at the Mining Company.

Resources

- The *Technology License and Distribution Agreement* is available for public viewing at http://java.sun.com/aboutJava/info/document.html
- "Sun goes public with new Java suit details" (*Sm@rt Reseller*) http://www.zdnet.com/pcweek/news/0706/09ajava.html
- "Sun Slams Microsoft on Java Again" (*Internet Week*) http://pubs.cmpnet.com/internetwk/news/news0709-4.htm
- "Sun and Microsoft in legal standoff" (*InfoWorld*) http://www.infoworld.com/cgi-bin/displayStory.pl?98079.wcsunms.htm
- "Sun Accuses Microsoft of More Java Incompatibilities" (*Computer Reseller News*) http://www.crn.com/dailies/weekending071098/jul09dig01.asp
- "New Microsoft Java flaws alleged" (CNet) http://www.news.com/News/Item/Textonly/0,25,24007,00.html